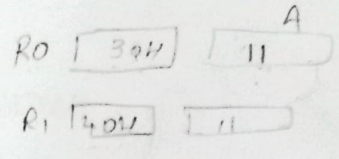
PROGRAM 1(a)

WALP to move a block of 10 bytes of data stored in the internal data memory from location 30H to the location starting from 40H of the internal data memory.

```
ORG 0000H
MOV R0, #30H
MOV R1, #40H
MOV R2, #10
BACK: MOV A, @ R0
MOV @R1, A
INC RO
INC R1
DJNZ R2, BACK
SJMP $
END
```



Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	i:40H	xx
31H	12h	41H	xx
32H	13h	42H	xx
33H	14h	43H	xx
34H	15h	44H	xx
35H	16h	45H	xx
36H	17h	46H	xx
37H	18h	47H	xx
38H	19h	48H	xx
39H	1ah	49H	Xx
Register	Data		
R0			
R1			
R2			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	i:40H	11h
31H	12h	41H	12h
32H	13h	42H	13h
33H	14h	43H	14h
34H	15h	44H	15h
35H	16h	45H	16h
36H	17h	46H	17h
37H	18h	47H	18h
38H	19h	48H	19h
39H	1ah	49H	1ah
Register	Data		
R0			
R1			
R2			

PROGRAM 1(b)

WALP to move a block of 10 bytes of data stored in the internal data memory to the external data memory.

Starting address of the internal data memory: 30H.

Starting address of the external data memory: 100H.

```
        ORG 0000H
        MOV R0, #30H
        MOV DPTR, #100H
        MOV R2, #10
BACK:   MOV A, @ R0
        MOVX @DPTR, A
        INC RO
        INC DPTR
        DJNZ R2, BACK
        SJMP $
        END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	X:100H	xx
31H	12h	101H	xx
32H	13h	102H	xx
33H	14h	103H	xx
34H	15h	104H	xx
35H	16h	105H	xx
36H	17h	106H	xx
37H	18h	107H	xx
38H	19h	108H	xx
39H	1ah	109H	xx
Register	Data		
R0			
DPTR			
R1			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	X:100H	11h
31H	12h	101H	12h
32H	13h	102H	13h
33H	14h	103H	14h
34H	15h	104H	15h
35H	16h	105H	16h
36H	17h	106H	17h
37H	18h	107H	18h
38H	19h	108H	19h
39H	1ah	109H	1ah
Register	Data		
R0			
DPTR			
R1			

PROGRAM 2 (a)

WALP to interchange the 10 bytes of data stored in the internal data memory from location 30H with 10 bytes of data stored from location 40H of the internal data memory

```
        ORG 0000H
        MOV R0, #30H
        MOV R1, #40H
        MOV R2, #10
BACK:   MOV A, @ R0
        XCH A,@R1
        MOV @R0, A
        INC RO
        INC R1
        DJNZ R2, BACK
        SJMP $
        END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	i:40H	aa
31H	12h	41H	bb
32H	13h	42H	cc
33H	14h	43H	dd
34H	15h	44H	ee
35H	16h	45H	ff
36H	17h	46H	99
37H	18h	47H	88
38H	19h	48H	77
39H	1ah	49H	66
Register	Data		
R0	30h		
R1	40h		
R2	0Ah		

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	aa	i:40H	11h
31H	bb	41H	12h
32H	cc	42H	13h
33H	dd	43H	14h
34H	ee	44H	15h
35H	ff	45H	16h
36H	99	46H	17h
37H	88	47H	18h
38H	77	48H	19h
39H	66	49H	1ah
Register	Data		
R0	3Ah		
R1	4Ah		
R2	00h		

PROGRAM 2(b)

WALP to interchange the 10 bytes of data stored in the internal data memory with the 10 bytes of external data memory location.

Starting address of the internal data memory: 30H.

Starting address of the external data memory: 100H.

```
ORG 0000H
 MOV R0, #30H
MOV DPTR, #100H
 MOV R2, #10
BACK: MOVX A, @ DPTR
 XCH A, @R0
 MOVX @DPTR, A
 INC RO
 INC DPTR
DJNZ R2, BACK
SJMP $
END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	11h	X:100H	aa
31H	12h	101H	bb
32H	13h	102H	cc
33H	14h	103H	dd
34H	15h	104H	ee
35H	16h	105H	ff
36H	17h	106H	99
37H	18h	107H	88
38H	19h	108H	77
39H	1ah	109H	66
Register	Data		
R0			
DPTR			
R1			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H	aa	X:100H	11h
31H	bb	101H	12h
32H	cc	102H	13h
33H	dd	103H	14h
34H	ee	104H	15h
35H	ff	105H	16h
36H	99	106H	17h
37H	88	107H	18h
38H	77	108H	19h
39H	66	109H	1ah
Register	Data		
R0			
DPTR			
R1			

PROGRAM 3(a)

WALP to add 10 bytes of binary/hex numbers stored in the internal data memory from location 30H. Store the 16 bit sum at location 40 and 41H such that MS byte of sum is stored at 40H of the internal data memory.

```
ORG 0000H
MOV R0, #30H
MOV R2, #10H
MOV R1, #00H
MOV A, R1
BACK:  ADD A, @ R0
JNC NEXT
INC R1
NEXT:  INC RO
DJNZ R2, BACK
MOV 40H, R1
MOV 41H, A
SJMP $
END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

PROGRAM 3(b)

WALP to add 10 bytes of BCD numbers stored in the internal data memory from location 30K. Stor the 16 bit sum at locations 40H and 41H such that MS byte of sum stored at 40H of the internal dat memory.

```
        ORG 0000H
        MOV R0, #30H
        MOV R2, #10H
        MOV R1, #00H
        MOV A, R1
BACK:   ADD A,@R0
        DAA
        JNC NEXT
        INC R1
NEXT:   INC R0
        BJNZ R2, BACK
        MOV 40H, R1
        MOV 41H, A
        SJMP $
        END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

PROGRAM 4(a)

WALP to add two multi byte numbers (at least 32 bits) stored in the internal data memory from location 30H and location 40H. Store the multi byte sum from location 50B of the internal data memory.

```
        ORG 0000H
        MOV R0, #30H
        MOV R1, #40H
        MOV R2, #50H
        MOV R3, #05H
        CLR C
BACK:   MOV A,@R0
        ADDC A,@R1
        MOV 04H, R1
        MOV R1, 02H
        MOV @R1, A
        MOV 02H, R1
        MOV R1, 04H
        INC R0
        INC R1
        INCR2
NEXT:   DJNZ R3, BACK
        MOV 01H, R2
        MOV A, #00H
        RLC A
        MOV @R1, A
        SJMP $
        END
```

Before execution

Source 1		Source 2		Destination	
Memory Address	Data	Memory Address	Data	Memory Address	Data
i:30H		i:40H		i = 50	XX
31H		41H		51H	XX
32H		42H		52H	XX
33H		43H			
34H		44H			
35H		45H			

After execution

Source 1		Source 2		Destination	
Memory Address	Data	Memory Address	Data	Memory Address	Data
i:30H		i:40H		i = 50	
31H		41H		51H	
32H		42H		52H	
33H		43H			
34H		44H			
35H		45H			

PROGRAM 4(b)

WALP to find the square and cube of an 8 bit binary number stored in the internal data memory location 30h

```
X EQU 30H
SQU EQU 31H
CUBE EQU 33H

ORG 0000H
MOV A,X
MOV R0,A
MOV B,A
MUL AB
MOV SQU,B
MOV SQU+1,A
MOV B,R0
MUL AB
MOV R1,A
MOV R2,B
MOV B,R0
MOV A,SQU
MUL AB
ADD A,R2
MOV R2,A
ADDC A,B
MOV R3,A
MOV CUBE,R3
MOV CUBE+1,R2
MOV CUBE+2,R1
SJMP $
END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:31H	XX
		32H	XX
		33H	XX
		34H	XX
		35H	XX

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:31H	XX
		32H	XX
		33H	XX
		34H	XX
		35H	XX

PROGRAM 5(a)

WALP to find the largest element of an array of 10 bytes of the data stored in the internal data memory from location 30h.

```
        ORG 0000H
        MOV R2,#10
        DEC R2
        MOV R0,#30H
        MOV A@R0
        MOV B,R0
        INC R0
BACK:   CLR C
        MOV R3,A
        SUBB A@R0
        JNC NEXT
        MOV 03H,@R0
        MOV B,R0
NEXT:   MOV A,R3
        INC R0
        DJNZ R2,BACK
        MOV 41H,R3
        MOV 40H,B
        SJMP $
        END
```

Before execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

After execution			
Source		Destination	
Memory Address	Data	Memory Address	Data
i:30H		i:40H	XX
31H		41H	XX
32H			
33H			
34H			
35H			
36H			
37H			
38H			
39H			

PROGRAM 5(b)

WALP to sort the array of 10 eight bit number stored in the internal data memory location from location 30h

```
        ORG 0000H
START:MOV R2,#10
      DEC R2
      MOV R0,#30H
      MOV R1,#31H
      MOV R3,#00H
BACK: CLR C
      MOV A,@R0
      SUBB A,@R1
      JC NEXT
      MOV A,@R0
      XCH A,@R1
      MOV @R0,A
      MOV R3,#01
      NEXT:INC R0
      INC R1
      DJNZ R2,BACK
      CJNE R3,#00,START
      SJMP $
      END
```

Before execution	
Source	
Memory Address	Data
i:30H	
31H	
32H	
33H	
34H	
35H	
36H	
37H	
38H	
39H	

After execution	
Destination	
Memory Address	Data
i:30H	
31H	
32H	
33H	
34H	
35H	
36H	
37H	
38H	
39H	

PROGRAM 6(a)

WALP to count the number of 1's in a byte which is be accepted from the port 0 and display the result in the port1.

```
        ORG 0000H
        MOV P0, #0FFH
BACK:   MOV A, P0
        MOV R0, #00H
        MOV R1, #08H
LOOP:   RLC A
        JNC NEXT
        INC R0
        NEXT: DJNZ R1, LOOP
        MOV P1, R0
        SJMP $
        END
```

Before Execution

Port-0	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
Port-1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

After Execution

Port-0								
Port-1							✓	✓

PROGRAM 6(b)

WALP to check whether the gives byte is a valid bit palindrome, accept the byte from the Port-0 and display **AA**h if valid else **00**H in the Port-1.

```
        ORG 0000H
        MOV P0,#0FFH
BACK:   MOV A,P0
        MOV B,A
        MOV R0,#08
        ACALL REVERSE
        MOV A,30H
        CJNE A,B,NEXT
        MOV P1,#0AAH
        SJMP LAST
NEXT:   MOV P1,#00H
LAST:   SJMP BACK

REVERSE: RLC A
         MOV R1,A
         MOV A,R2
         RRC A
         MOV R2,A
         MOV A,R1
         DJNZ R0,REVERSE
         MOV 30H,R2
         RET
         END
```

Before Execution

Port-0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
Port-1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

After Execution

Port-0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Port-1	✓		✓		✓		✓	

PROGRAM 7(a)

WALP to convert a two digits DECIMAL/8 bit BCD number to its equivalent 2 digit HEX /8 bit binary number.

```
ORG 0000H
MOV P0,#0FFH
MOV A,P0
MOV B,#10H
DIV AB
MOV R0,B
MOV B,#10
MUL AB
ADD A,R0
MOV P1,A
SJMP $
END
```

Before Execution

Port-0	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**
Port-1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

After Execution

Port-0	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**
Port-1					✓	✓	✓	✓

PROGRAM 7(b)

WALP to convert a two digits HEX/8 bit binary number to its binary number to its equivalent t... digit DECIMAL/8bit BCD number.

```
ORG 0000H
MOV P0,#0FFH
MOV A,P0
MOV B,#10
DIV AB
MOV R0,B
MOV B,#10
DIV AB
MOV R2,A
MOV R1,B
MOV A,R1
SWAP A
ADD A,R0
MOV P1,R2
MOV P2,A
SJMP $
END
```

Before Execution

Port-0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
Port-1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

After Execution

Port-0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
Port-1				✓		✓		✓

PROGRAM 8(a)

WALP to convert an 8 bit BCD/2 digit DECIMAL number to its equivalent ASCII number.

```
ORG 0000H
MOV P0,#0FFH
MOV A,P0
MOV B,#10H
DIV AB
ORL A,#30H
MOV P1,A
ORL B,#30H
MOV P2,B
SJMP $
END
```

Before Execution

Port-0	0	1	0	1	0	0	1	1
Port-1	X	X	X	X	X	X	X	X
Port-2	X	X	X	X	X	X	X	X

After Execution

Port-0	0	1	0	0	0	0	1	1
Port-1			✓	✓		✓		✓
Port-2			✓	✓			✓	✓

PROGRAM 8(b)

WALP to convert the given two bytes of ASCII number to its equivalent 8 bit BCD/2 digit DCIMAL number.

```
ORG 0000H
MOV P0,#0FFH
MOV A,P0
MOV B,#10H
DIV AB
ORL A,#30H
MOV P1,A
ORL B,#30H
MOV P2,B
SJMP $
END
```

Before Execution

Port-0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**
Port-1	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
Port-2	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

After Execution

Port-0	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**
Port-1	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**
Port-2		✓		✓			✓	✓

PROGRAM 9

The output the value AAH and 55H alternatively to the port 0 for every 2 sec.

```
        ORG 0000H
        MOV TMOD,#10H
        MOV A,#55H
L1:     MOV P0,A
        ACALL DELAY
        CPL A
        SJMP L1

        ORG 0050H
DELAY:  MOV R0,#40
L3:     MOV TH1,#3CH
        MOV TL1,#0B0Hd
        SETB TR1
L2:     JNB TF1,L2
        CLR TR1
        CLR TF1
        DJNZ R0,L3
        RET
        END
```

PROGRAM 10

WALP to transfer the string of data using serial communication by configuring the serial port in mode_1 with a baud rate of 9600,1 step bit and 8 data bits.

```
        ORG 0000H
        MOV TMOD,#20H
        MOV TH1,#-3
        MOV SCON,#50H
        SETB TR1
        MOV DPTR,#DATA1
START:  CLR A
        MOVC A,@A+DPTR
        JZ STOP
        MOV SBUF,A
L1:     JNB T1,L1
        CLR T1
        INC DPTR
        SJMP START
STOP:   NOP

        ORG 100H
DATA1:DB "DEPARTMENT OF ECE",0;
        END
```

PROGRAM 11

WALP to count the number of inputs to the counter_0 by configuring the timer as counter in mode_1 and display the count in ports.

```
        ORG 0000H
        MOV TMOD,#05H
        SETB P3.5
L1:     MOV TL0,#0
        MOV TH0,#0
        SETB TR0
L2:     MOV P2,TL0
        MOV P1,TH0
        JNB TF0,L2
        CLR TR0
        CLR TF0
        SJMP L1
        END
```

PR0GRAM 12

Let X,Y,Z refers to the contents of the memory location 30h,31h,and 32h respectively write an
to perform the following logical operations;
If X=00 perform the operation Y OR Z.
If X=01 perform the operation Y AND Z.
If X=02 perform the operation Y XOR Z.

```
        ORG 0000H
        MOV R1,30H
        MOV A,31H
        MOV B,32H
        CJNE R1,#00H,BRAND
        ORL A,B
        SJMP LAST
BRAND:  CJNE R1,#01H,BRXOR
        ANL A,B
        SJMP LAST
BRXOR:  CJNE R1,#02H,LAST
        XRL A,B
LAST:   MOV 33H,A
        SJMP $
        END
```

PROGRAM 13

WALP to compare the bytes of data present at the memory location 21h and 22h and represent the result of comparison through the bits whose addresses are 00h and 01h.
If (21h)<(22h)then clear the bit at 01h and also set the bit at 00h.
If (21h)>(22h)then set the bit at 01h and also clear the bit at 00h.
If (21h)=(22h)then set the bit at 01h and also set the bit at 00h.

```
        ORG 0000H
        MOV A, 21H
        CLR C
        SUBB A, 22H
        JZ EQUAL
        JNC BIG
        SETB 00H
        CLR 01H
        SJMP LAST
BIG:    CLR 00H
        SETB 01H
        SJMP LAST
EQUAL:  SETB 00H
        SETB 01H
LAST:   SJMP $
        END
```

Case 1:

Before execution	
Source	
Memory Address	Data
i:20H	*XX*
21H	*15*
22H	*14*

After execution	
Destination	
Memory Address	Data
i:20H	*01*
21H	*15*
22H	*14*

Case 2:

Before execution	
Source	
Memory Address	Data
i:20H	*XX*
21H	*14*
22H	*15*

After execution	
Destination	
Memory Address	Data
i:20H	*02*
21H	*14*
22H	*15*

Case3:

Before execution	
Source	
Memory Address	Data
i:20H	*XX*
21H	*15*
22H	*15*

After execution	
Destination	
Memory Address	Data
i:20H	*03*
21H	*15*
22H	*15*